A heterogeneous multiprocessor architecture for low-power audio signal processing applications

Paker, Ozgun; Sparsø, Jens; Haandbæk, Niels; Isager, Mogens; Nielsen, Lars Skovby

Published in:
Proceedings on IEEE Computer Society Workshop

Link to article, DOI:
10.1109/IWV.2001.923139

Publication date:
2001

Document Version
Publisher's PDF, also known as Version of record

A Heterogeneous Multiprocessor Architecture for Low-Power Audio Signal Processing Applications

Özgün Paker[1] Jens Sparsø[1] Niels Haandbæk[2] Mogens Isager[2] Lars Skovby Nielsen[2]

[1] Informatics and Mathematical Modeling
Technical University of Denmark
2800 Lyngby, Denmark
{opa,jsp}@imm.dtu.dk

[2] Oticon A/S, Strandvejen 58
2900 Hellerup, Denmark
{nch,moi,lsn}@oticon.dk

Abstract

This paper describes a low-power programmable DSP architecture that targets audio signal processing. The architecture can be characterized as a heterogeneous multiprocessor consisting of small and simple instruction set processors called mini-cores that communicate using message passing.

The processors are tailored for different classes of filtering algorithms (FIR, IIR, N-LMS etc.), and in a typical system the communication among processors occur at the sampling rate only. The processors are parameterized in word-size, memory-size, etc. and can be instantiated according to the needs of the application at hand using a normal synthesis based ASIC design flow. To give an impression of the size of a processor we mention that one of the FIR processors in a prototype design has 16 instructions, a 32 word × 16 bit program memory, a 64 word × 16 bit data memory and a 25 word × 16 bit coefficient memory.

Early results obtained from the design of a prototype chip containing filter processors for a hearing aid application, indicate a power consumption that is an order of magnitude better than current state of the art low-power audio DSP's implemented using full-custom techniques. This is due to: (1) the small size of the processors and (2) a smaller instruction count for a given task.

1. Introduction

The growing demand for increasingly sophisticated portable computing and communication devices is the main driving force behind research in low-power VLSI design. The design of these future single-chip, full-function processing devices require flexibility and re-usability which calls for programmable processor-based solutions. Unfortunately programmability and energy-efficiency are conflicting goals as illustrated in figure 1: dedicated hardwired circuits (ASICs) offer low-power consumption, high speed, and small area but they are not flexible. Even a small change in function calls for a redesign and the fabrication of a new chip. At the other end of the spectrum are DSP's, and general purpose microprocessors (μP).

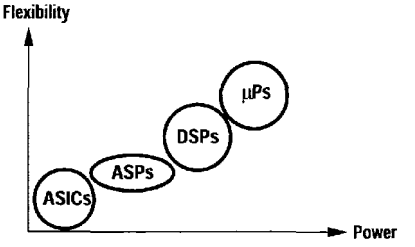

Figure 1. Power versus flexibility.

Ideally one would want the power efficiency of a hardwired ASIC solution while maintaining the flexibility of a programmable processor, and the "territory" between the hardwired ASICs and the general purpose DSP's attracts a significant amount of research interest [2, 11, 13, 4, 6]. One camp attacks the problem from the DSP side and advocates so-called *ASPs – application specific processors;* i.e. specialized instruction set processors that are optimized for a given set/domain of algorithms. Another camp attacks the problem from the ASIC-side and provides the designer/programmer with a set of RTL-level components (register files, multipliers, adders etc.) and a (dynamically) reconfigurable network that allow arbitrary data-flow types of computing structures to be formed.

From a hardware implementation perspective these solutions can be characterized as coarse grained and fine grained respectively. In this paper we propose an architecture that falls between the two, but with an ASP flavor.

The application domain we are considering: audio signal processing – and more specifically digital hearing aids;

0-7695-1056-6/01 $10.00 © 2001 IEEE

is characterized by extremely low power consumption requirements. Total power consumption in the order of 0.5 - 1.0 mW (at 1.0 V supply) is typical. For this reason, many commercial hearing aids are based on hardwired ASIC solutions (including the recently published [7]) but fully programmable DSP-based solutions are also starting to emerge [6].

The work described in this paper is an attempt to develop a programmable platform whose energy efficiency approaches that of a dedicated ASIC. The work is done at DTU as part of a formal collaboration between Oticon Inc. and DTU on low power signal processing. The contributions of the 3rd and 4th author relates to their M.Sc. projects at DTU. The purpose of this paper is to present the architecture, to elaborate on the design and implementation of a prototype, and to evaluate the power consumption and to compare it with some (published) alternatives. The specific test-chip that we present includes some representative parts of a hearing aid; but the test-chip is not a complete solution. The purpose of designing and implementing a test-chip is to prove the concept and to obtain confidence in the area, speed and power figures estimated by our CAD tools.

The paper is organized as follows. Section 2 presents the baseline for our work and the overall architecture of the proposed solution. Section 3 puts the proposed architecture in perspective by discussing some related work. Sections 4 and 5 elaborate on the architecture and its implementation. Section 6 presents a prototype chip that has been designed to evaluate the concept. Section 7 presents the power consumption results and compares with existing low-power audio DSP's. Finally section 8 concludes the paper.

2. Overall architecture

The design of an audio signal processing application (as for example a hearing aid) usually starts with a specification in Matlab – often in the form of a complex Simulink data-flow structure of filters and other signal processing blocks that communicate at the sampling rate: FIR, IIR, N-LMS, Viterbi, FFT, etc. The idea is to provide a platform composed of simple instruction set processors called mini-cores each optimized for one of these classes of algorithms, and to provide a communication network that supports message passing among processors as shown in figure 2.

Because the mini-cores are small and specialized, because the necessary data structures are kept locally inside the processors, and because the processors communicate at moderate rates (as compared with the internal clock rate) such an implementation will potentially be very power efficient. The other desired feature - flexibility - is provided through a multitude of different processors. Furthermore such an architecture is inherently modular: It is a simple task to add new mini-cores, and the message passing approach to

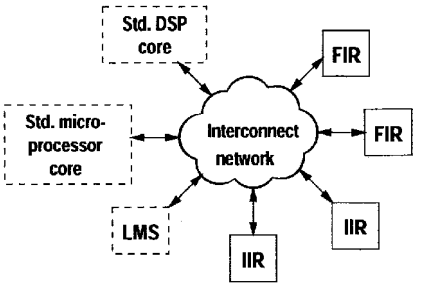

Figure 2. The mini-core system architecture.

communication, makes it a simple task to fit in general purpose microprocessor- and DSP cores as well.

Designing a mini-core based platform for a given application involves instantiating different processors as well as different versions of some of the processors. To enable this we envision a traditional synthesis-based ASIC design flow, where (parameterized) VHDL descriptions of the different processors are mapped into netlists of standard cells. This soft-macro approach has further advantages: (1) it allows the integration of other proprietary circuits on the same chip, and (2) the implementation is foundry independent.

The general trend in design of portable battery powered applications is that low power consumption is the main concern, while area and in particular speed are less of a concern. The proposed architecture is in line with this. It consists of a multitude of relatively small dedicated processors. The processors may not be active all the time and if the same platform is used in different products there may even be unused processors. For this reason, the processors are designed to have zero dynamic power consumption when idle.

3. Related work

A related approach is taken in the Pleiades project [2, 11, 13]. Here an on-chip general purpose microprocessor (ARM8) is augmented with an array of heterogeneous programmable units called "satellite processors" that are connected by a reconfigurable network. While the microprocessor supports the control-intensive components of the applications, repetitive and regular data-intensive loops are directly mapped onto the array of satellites by downloading some parameters and by configuring the interconnections between them. A typical satellite processor in this approach would be a multiply-accumulate unit, a memory or an address generator, and the configuration of the satellite processors corresponds to wiring up a dedicated data-flow circuit. To accommodate the need for non-numerical computations the chip also has a block of traditional fine-grain FPGA logic. Because the communication rate between the satellite processors is rather high - typically close to the clock rate,

the interconnection network is highly optimized, exploiting low-swing full-custom circuitry [14]. In this respect our approach is different: the stored-program instruction set processors keep data structures and operator modules local, and the inter processor communication typically occur at a rate that is close to the sampling rate.

Another related work is [4] where an instruction set processor with a configurable datapath is presented. The application domain covers various wireless communication standards. The datapath basically consists of simple functional units: multipliers, ALUs and shifters. The instruction set of this architecture can be extended with macro-operations that can configure a compound computational unit using the basic functional units. These macro-operations look like the LMS and FIRS instructions found in the TMS320C54x DSP processor. The output of any functional unit can be input to another by a configurable feedback path. In our approach, we also have compound functional units to decrease the instruction count of sophisticated DSP algorithms, but we avoid the complexity of configurable structures. For instance, a *dedicated* dual-multiply-accumulate unit exists in the IIR mini-core in order to handle biquad filters efficiently.

Finally, it is relevant to mention a couple of state of the art low-power DSP's intended for audio applications. The designs presented in [8] and [5] all use a variety of full-custom circuit techniques, and some of them even use dual V_t processes to obtain high speed and low standby power consumption at the same time. The Coyote processor developed by GN Resound and Audiologic is among the most power efficient designs in existence today [6, 1]. This design significantly resembles a general purpose DSP architecture with optimizations that emphasize audio signal processing. It has a specialized instruction set and a datapath with a special multiply accumulate unit called PMAC. Compared with our approach it is a much more coarse grained processor, and when it comes to power efficiency it benefits from its full-custom implementation and (like any other traditional general purpose DSP) it suffers from its size and from its generality.

4. Mini-cores

This section gives an overview of the design philosophy for mini-cores and present two particular implementations.

4.1. Introduction

Each mini-core has a customized datapath for a particular class of algorithm. For instance the FIR mini-core employs an add-multiply-accumulate unit for linear phase filters. Likewise the IIR mini-core has a dual-multiply-accumulate unit to handle biquad sections in fewer clock cycles. On top of these customized datapaths, the instruction sets of both mini-cores are extremely simple and small.

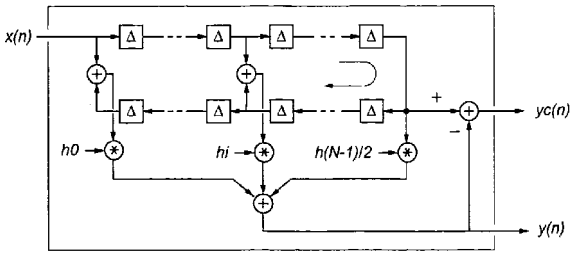

Figure 3. An interpolated FIR filter used in hearing aids.

The mini-cores are initialized before run-time by downloading programs and data constants as well as routing parameters to the interconnect network. The initialization is done using a configuration bus not shown in figure 2. The bus effectively maps all register files and all data, constant, and program memories into one random access address space. The configuration bus can also be used for testing purposes.

4.2. The FIR mini-core

The computations done by the FIR mini-core consists of a series of multiply-accumulate steps to compute the output of a FIR filter. The operands are filter coefficients and samples stored in a delay-line. For audio applications it is often desirable to use linear-phase filters. The coefficients for such a filter are symmetric around the midpoint of the impulse response. A linear-phase filter can thus be implemented efficiently by a folded structure, where two samples from the delay-line are added before being multiplied with the corresponding coefficient. Interpolated filters are often used in filter banks. An interpolated filter has a large number of absent taps. Figure 3 shows an example of a linear phase interpolated FIR filter that we have studied in previous work [9, 10]. The filter in figure 3 produces two outputs that have symmetry with respect to the half-band frequency $\pi/2$.

```
switch  clr, 0        ; Switch to filter 0
receive tmp1, 0       ; Receive sample
mov     regdm, tmp1   ; Move sample to delay-line
asmacc  clr, 2        ; First macc with
asmacc  2             ; clear of macc register.
asmacc  1
macc    comp, fin, 5; Final macc. Generate
                      ; complementary output and
                      ; adjust pointers.
nop                   ; Delay-slot.
send    macc, 10      ; Send output...
send    comp, 20      ; and complementary output.
```

The FIR mini-core has instructions dedicated to ordinary filters (`macc`) and linear-phase filters (`asmacc`). These instructions automatically update the pointers into the delay-

line and the list of coefficients and are thus very powerful for this particular application. The mini-core can also switch between filters using a single instruction (`switch`). The above program implements a small linear-phase filter of length 11 with seven non-zero coefficients. The numeric arguments to the `macc` and `asmacc` instructions are used to skip taps with zero coefficients.

Filters implemented on the FIR mini-core typically use significantly fewer instructions per sample as compared to a DSP processor. This is mainly due to the mini-core being able to skip coefficients with taps that are zero, but also because the overhead associated with switching from one filter to the next is only a single instruction. Therefore, a filter implemented on a FIR mini-core has a much lower instruction count than a traditional DSP implementation.

The datapath for the FIR mini-core is shown in figure 4. As seen in the figure, the mini-core has four different memories, each with a specific purpose. The delay-line memory has two read ports, which are used in linear-phase filter implementations. The filter memory holds the parameters that specify a FIR filter. These are: the length of the filter; a pointer to the base of a coefficient array; an index into the coefficient array; a pointer to the base of the delay-line; and two indices into the delay-line. The datapath is a simple two-stage pipeline. The first stage handles the fetch and decode of instructions and the second stage handles the multiply-accumulate computation.

4.3. The IIR mini-core

High order IIR filters are usually realized by a serial and/or parallel combination of low order IIR filters, to alleviate coefficient quantization sensitivity of the filter in question. The basic element for implementing a high order IIR filter is a second order IIR filter of direct form II implementation as shown in figure 5, known as a "biquad."

The IIR mini-core has a dual-multiply-accumulate unit that enables the computation of an entire biquad section in two clock cycles. Basically it is a combinational unit that computes two multiplications and adds both results of the multiplications and the accumulator.

Another feature of the IIR mini-core that differentiates

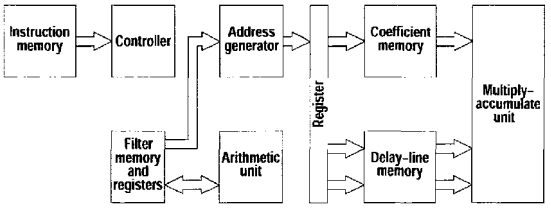

Figure 4. Block diagram of the FIR mini-core.

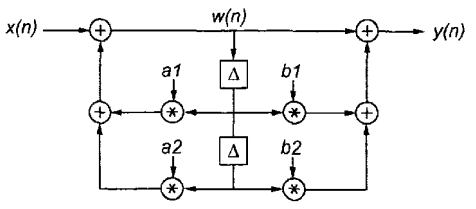

Figure 5. A biquad section.

it from a DSP processor is the specialized register file used to store the delay elements of a biquad section. As shown in figure 6 the register file is implemented as a set of two-word shift-registers. The address input controls which register pair to read. The shift-register pair implementation of a biquad stage is a direct mapping of the delay elements in figure 5 to hardware.

The IIR datapath is shown in figure 7. The design has separate memories for storing programs and coefficients. The shift-add unit in the datapath is used for scaling input and/or output as well as implementing biquad sections that has a small number of '1's in their coefficients.

The IIR mini-core executes a biquad section, using a single instruction called `biq`. The following program shows a 4^{th} order cascaded IIR filter implementation.

```
receive r1, 0      ; Receive sample to r1
biq     w0,r2,r1   ; Compute biquad 1,
biq     w1,r3,r2   ; Compute biquad 2,
send    r3, 4      ; Send output...
```

5. Interconnect network

The mini-cores communicate over a network using message passing supported by `send` and `receive` instructions. Only point-to-point channels are supported.

Each mini-core has a number of input buffers, corresponding to the input channels, and an output buffer that is shared by the output channels. Depending on its topology

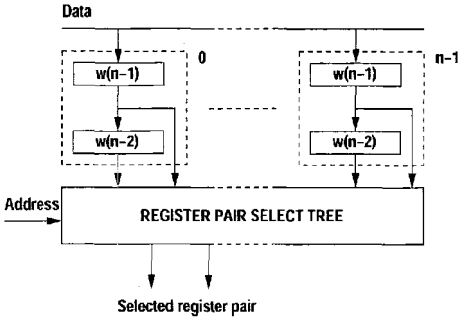

Figure 6. Register file implementation.

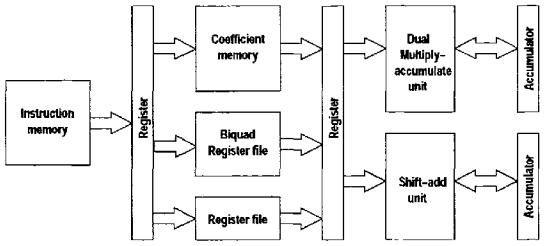

Figure 7. Block diagram of the IIR mini-core.

the network may provide additional buffering. Because of this buffering the exchange of a message does not synchronize the two communicating mini-cores. A mini-core executing a `receive` instruction goes to "sleep" until the requested data item shows up at the specified channel. Likewise a mini-core executing a `send` instruction halts until the network consumes the data item in the output buffer. These sleeping modes are handled by clock gating at the module level. A mini-core is only clocked when necessary, and this results in significant power savings.

The send and receive buffers described above are part of an "interface" module as shown in figure 8. The interface module provides an abstraction level to the mini-core designer, and separates the design and topology of the interconnection network from the mini-core.

The choice of interconnect should be made based on the communication requirements of the system. For a few mini-cores that communicate few messages in a sample period, a simple bus structure is often sufficient. If the number of nodes is increased the limited bandwidth and in particular the capacitive load of the shared bus may become a problem. To solve this, an interconnect structure that allows several simultaneous transfers such as a torus network may be used. In addition to the improved bandwidth, the torus also has a well-balanced capacitive load distribution to all nodes and

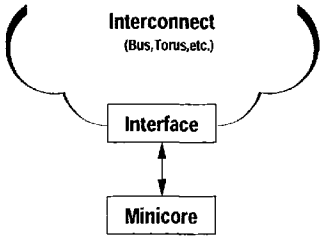

Figure 8. The mini-core is connected to the nodes of the interconnect structure via an interface module.

therefore represents a good power efficient solution. We are currently working on such scalable interconnect structures.

The test chip described in the next section has a bus based interconnect network with a simple round robin arbitration scheme (based on a circulating token).

6. Test chip

To assess the feasibility of our approach, we designed a test chip with 6 mini-cores, 3 FIR, and 3 IIR. The mini-cores are instantiated with different memory sizes and will run parts of a non-trivial industrial application. Table 1 shows some of the parameters that are set during mini-core instantiation. Typical filter examples are used to determine these parameters. The Data Memory field for FIR type mini-cores shows the sizes of delay-line memories whereas for IIR type mini-cores, that field represents the sizes of biquad register files.

To measure the power consumption of the network and the mini-cores individually, separate power and ground pins are added to the pin configuration. The communication network is also available at the pins, allowing a possible extension to the number of mini-cores outside the chip. For instance, even a micro-controller could be added to the system.

A layout of the test chip is shown in figure 9. The test chip is implemented using STMicroelectronics standard cell library with a minimum feature size of $0.25\mu m$. The core area is approximately $5mm^2$ and contains 520 K transistors.

7. Results

We have evaluated the power consumption of the mini-cores by simulating the gate-level netlists. The resulting switching-activity information was then used by the Synopsys toolset, to estimate the power consumption. Our pre-layout estimations are based on statistic wire load models. A previous design experience showed a $\pm 10\%$ discrepancy between wire load and RC back-annotated post-layout estimations. The simulations are done at 1.8 Volt supply volt-

Mini-core	Data Memory words x bits	Instruction Memory words x bits	Coefficient Memory words x bits
FIR1	118x16	41x16	16x16
FIR2	93x16	32x16	16x16
FIR3	64x16	32x16	25x16
IIR1	4x20	32x13	4x12
IIR2	8x20	32x14	8x12
IIR3	16x25	64x15	16x20

Table 1. Mini-core parameters.

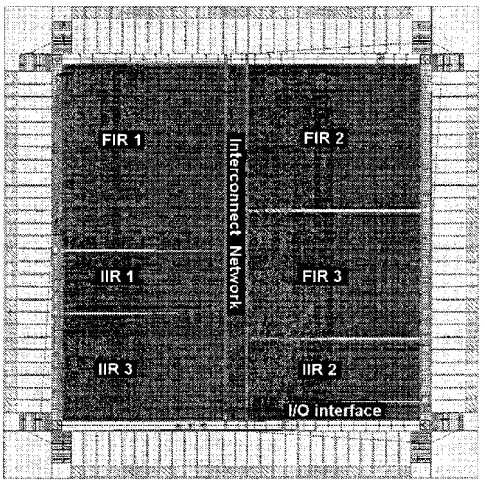

Figure 9. Floorplan of the test chip.

FIR filter	FIR mini-core	TI-C54x
Inst. per sample:	30	106
Clock frequency:	480 kHz	1696 kHz
Power for task @1V:	48.5 μW	621 μW
Power per MIPS	101 μW/MHz	366 μW/MHz
IIR filter	IIR mini-core	TI-C54x
Inst. per sample:	12	59
Clock frequency:	192 kHz	994 KHz
Power for task @1V:	10.5 μW	251 μW
Power per MIPS	54 μW/MHz	265 μW/MHz

Table 2. Power consumption of different filter implementations assuming a 16 KHz sampling rate. The figures for the FIR mini-core and the IIR mini-core can be compared with similar figures for a TI-C54x DSP. All figures assume a supply voltage of 1.0V.

age. As we target operation at 1 Volt we have scaled the results.

Table 2 lists the power consumption of two individual minicores: an FIR mini-core running a filter of length 64 with 25 non-zero coefficients, and an IIR mini-core running two biquad sections. We assume a sampling frequency of 16 kHz, and the clock rates stated in table 2 is simply "instruction count" × "sampling clock" – i.e., the minimum possible clock rate that will allow the mini-cores to cope with a 16 kHz stream of samples. The Power per MIPS figures relate to the discussion about alternative DSP designs later in this section.

Texas Instruments have documented the power performance of their TMS320C54x family of processors in [12]. Using information from this document combined with sample programs from Texas Instruments [3], we have estimated the instruction count and power consumption for a C54x executing the same FIR and IIR filters at the same 16 kHz sampling rate. The estimation is fairly rough, but at least it gives an indication of how a mini-core performs compared to a DSP processor (that is also fabricated in a 0.25 μm CMOS process). The measurements in [12] are done at a supply voltage of 3 Volt, and we have scaled them to 1V to enable comparison, table 2.

As seen from table 2 the FIR mini-core consumes only 7.8 % of the power consumed by the C54x. This huge power saving is due to two factors: (1) the FIR mini-core requires 3.5 times fewer instructions for the same task, and (2) each FIR mini-core instruction consumes 3.6 times less energy (despite the fact that it does more). For the IIR mini-core the picture is even more favourable. Its power consumption is only 4.2 % of the corresponding figure for the C54x. It executes 4.9 times fewer instructions and has a 4.7 times lower energy/instruction figure.

In summary the absolute power efficiency of the mini-cores is around 50-100μW/MIPS (for relatively complex instructions), and in relative terms the power efficiency of the mini-core approach is 15-20 times better than the TI-C54x.

This discussion illustrates the difficulty of making a fair comparison of alternative designs. Many articles on low power DSP architectures only state energy-per-instruction measures like: MIPS/W (mega instructions per second per Watt), MOPS/W (mega operations per second per watt) or MMACs/W (mega multiply-accumulates per second per Watt). These figures should be taken with some care as they completely ignore the instruction-count-per-task issue. A fair comparison requires one or more real benchmarks for which one can estimate the energy consumption. Unfortunately such information is generally unavailable, and for a comparison with more recent designs aiming specifically for low power we have to restrict to the energy-per-instruction figures.

The Coyote DSP processor [6] is specifically designed for audio signal processing and low power consumption. It was originally implemented in a 0.50 μm CMOS process, but it has been re-implemented in 0.25 μm technology where it consumes 100 $\mu W/MIPS$ [1]. The authors of [5] achieve 210μW/MHz in a 0.35 μm dual V_t CMOS technology. The benchmark application consists of mainly MAC instructions. A 0.5 μm multi-threshold CMOS DSP by [8] offers 1.1 mW/MHz while running MACs. All these designs involve full-custom layout, and can be characterized as "optimized" DSP's where an instruction typically involves one multiply-accumulate operation and some address pointer updating. Furthermore they all owe a great deal of their power efficiency to low-level full-custom circuit implementations.

To complete the picture we mention that an implementation of the Pleiades architecture achieves 10-100MOPS/mW [13], corresponding to 10-100 μW/MOPS, and that a hardwired fully synthesized hearing aid IC achieves 14.4 μW/MOPS [7]. For these designs it is rather unclear what is meant by an "instruction" or an "operation," and it is therefore unclear how to compare with our design. The mini-cores consume approximately 50-100 μW/MHz but they execute 3-4 times fewer instructions than a traditional DSP for the same task (table 2) hinting that 15-30 μW/MOPS is perhaps more realistic for comparison with [13, 7].

Furthermore, it is important to stress that the mini-core design presented here is a first attempt – its purpose is to prove the concept. The prototype chip is 100 % synthesized, uses latch-based memories, standard Synopsys designware multipliers and adders, and it is implemented in a standard cell library that is not optimized for low power. We believe that by re-implementing the design in a low power standard cell library and by optimizing the multiply-accumulate and memory modules in both designs, it is possible to reduce the power consumption by a factor of two.

8. Conclusion

This paper presented a low-power programmable DSP architecture that targets audio signal processing. The application domain covers different classes of filtering algorithms (FIR, IIR, N-LMS etc.). The architecture is a heterogeneous multiprocessor consisting of small and simple instruction set processors that communicate using message passing. These processors called mini-cores are tailored to particular classes of algorithms from the application domain. The mini-cores are parameterized in word-size, memory-size, etc. and can be instantiated according to the needs of the application at hand using a normal synthesis based ASIC design flow.

Early results obtained from the design of a prototype chip indicate a power consumption that is an order of magnitude better than current state of the art low-power audio DSP's implemented using full-custom techniques. This is due to: (1) the small size of the mini-cores and (2) a smaller instruction count for a given task.

When comparing with a state of the art hardwired synthesized hearing aid IC [7] or the dynamically reconfigurable Pleiades architecture [13] implemented using highly optimized circuitry including low swing interconnect, the mini-core architecture has a comparable but slightly higher power consumption. However, as indicated in the previous section, we expect to optimize the design further. In addition it should be stressed that the mini-core platform is both programmable and implemented using standard cells and a synthesis based design flow.

The work presented here represents a first step. Future work will include implementing an LMS mini-core and adding a general purpose microprocessor or DSP such that a full hearing aid application can be implemented and compared with an existing solution. Future work also includes tools for mapping applications onto a mini-core based platform.

References

[1] http://www.audiologic.com.

[2] A. Abnous and J. Rabaey. "Ultra-Low-Power Domain-Specific Multimedia Processors". In *Proceedings of the IEEE VLSI Signal Processing Workshop*, pages 461–470, October 1996.

[3] Optimized DSP Library for C Programmers on the TMS320C54x. Application report, Texas Instruments, January 2000.

[4] T. A. Lee, D. C. Cox, J. Nichols, and S. Asghar. "Low Power Reconfigurable Macro-Operation Signal Processing for Wireless Communications". In *48th IEEE Vehicular Technology Conference*, volume 3, pages 2560–2564, May 1998.

[5] W. Lee and et al. "A 1-V Programmable DSP for Wireless Communications". *IEEE Journal of Solid State Circuits*, 32(11):1766–1776, November 1997.

[6] F. Møller, N. Bisgaard, and J. Melanson. "Algorithm and Architecture of a 1 V Low Power Hearing Instrument DSP". In *International Symposium on Low Power Electronics and Design*, pages 7–11, August 1999.

[7] P. Mosch, G. V. Oerle, S. Menzl, N. Rougnon-Glasson, K. V. Nieuwenhove, and M. Wezelenburg. "a 720 μW 50 MOPs 1V DSP for a Hearing Aid Chip Set". In *Proceedings ISSCC 2000*, pages 238–239, Feb. 2000.

[8] S. Mutoh and et al. "A 1-V Multithreshold-Voltage CMOS Digital Signal Processor for Mobile Phone Application". *IEEE Journal of Solid State Circuits*, 31(11):1795–1802, November 1996.

[9] L. S. Nielsen and J. Sparsø. An 85 μW Asynchronous Filter-Bank for a Digital Hearing Aid. In *Proc. IEEE International Solid State circuits Conference*, pages 108–109, 1998.

[10] L. S. Nielsen and J. Sparsø. Designing asynchronous circuits for low power: An IFIR filter bank for a digital hearing aid. *Proceedings of the IEEE*, 87(2):268–281, Feb. 1999. Special issue on "Asynchronous Circuits and Systems" (Invited Paper).

[11] J. Rabaey. "Reconfigurable Computing: The Solution to Low Power". In *Proceedings 1997 ICASSP Conference*, April 1997.

[12] C. Turner. "Calculation of TMS320LC54x Power Dissipation". Application report, Texas Instruments, 1997. http://www-s.ti.com/sc/psheets/spra164/spra164.pdf.

[13] H. Zhang, V. Prabhu, V. George, M. Wan, M. Benes, A. Abnous, and J. Rabaey. "A 1-V Heterogenous Reconfigurable DSP IC for Wireless Baseband Digital Signal Processing". *IEEE Journal of Solid State Circuits*, 35(11):1697–1704, November 2000.

[14] H. Zhang, M. Wan, V. George, and J. Rabaey. "Interconnect Architecture Exploration for Low Energy Reconfigurable Single-Chip DSPs". In *IEEE Computer Society Workshop On VLSI'99*, pages 2–8, April 1999.